GPU Accelerated Newton for Taylor Series Solutions of Polynomial Homotopies in Multiple Double Precision

Jan Verschelde[†]

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
http://www.math.uic.edu/~jan
https://github.com/janverschelde
janv@uic.edu

2024 SIAM Conference Parallel Processing for Scientific Computing, CP6 Performance and Scalability

[†]Supported by the National Science Foundation, DMS 1854513 and a 2023 Simons Travel Award.

Outline

1. Problem Statement
   - numerical analytic continuation
   - the need for multiple double precision
   - software packages QDlib, GQD, and CAMPARY
   - convergence and performance

2. Newton for Taylor Series
   - monomial and binomial homotopies
   - linearization of power series
   - staggered computations
   - computational results

scalable polynomial homotopy continuation

towards a scalable polynomial system solver

With homotopy continuation we can track millions of solution paths for systems of polynomial equations in modest dimension.

Goal: track a solution path defined polynomial homotopies in large dimensions, at least 1024 equations and variables.

A polynomial homotopy is a system of polynomials in one parameter t, the solution trajectories are then also analytic functions in t, therefore *apply analytic continuation* to approximate the solutions.

Graphics Processing Units capable of teraflop performance can compensate for the cost overhead caused by quad double arithmetic.

The problems become *compute bound* already with complex double double and quad double arithmetic.

error free computations

A *multiple double* is an unevaluated sum of nonoverlapping doubles.

Take 64 random complex numbers on the unit circle.
The 2-norm of this vector is 8, computed with multiple doubles:

```
double double : 8.00000000000000E+00 - 4.46815747097839E-32
  quad double : 8.00000000000000E+00 + 8.23258305145073E-65
  octo double : 8.00000000000000E+00 - 5.56764060802733E-128
  hexa double : 8.00000000000000E+00 - 1.54394135726410E-257
```

Cost overhead:

	add	mul	div	avg
2	20	23	70	37.7
4	89	336	893	439.3
8	269	1742	5126	2379.0
16	925	11499	33041	15155.0

power series arithmetic

motivation for multiple double precision

$$\exp(t) = \sum_{k=0}^{d-1} \frac{t^k}{k!} + O(t^d).$$

Assuming the quadratic convergence of Newton's method:

k	$1/k!$	recommended precision	`eps`
7	`2.0e-004`	double precision okay	`2.2e-16`
15	`7.7e-013`	use double doubles	`4.9e-32`
23	`3.9e-023`	use double doubles	
31	`1.2e-034`	use quad doubles	`6.1e-64`
47	`3.9e-060`	use octo doubles	`4.6e-128`
63	`5.0e-088`	use octo doubles	
95	`9.7e-149`	need hexa doubles	`5.3e-256`
127	`3.3e-214`	need hexa doubles	

`eps` is the multiple double precision

software for multiple double arithmetic

- QDlib by Y. Hida, X. S. Li, and D. H. Bailey.
  Algorithms for quad-double precision floating point arithmetic.
  In the *Proceedings of the 15th IEEE Symposium on Computer Arithmetic*, pages 155–162, 2001.
- GQD by M. Lu, B. He, and Q. Luo.
  Supporting extended precision on graphics processors. In the *Proceedings of the Sixth International Workshop on Data Management on New Hardware (DaMoN 2010)*, pages 19–26, 2010.
- CAMPARY by M. Joldes, J.-M. Muller, V. Popescu, and W. Tucker.
  CAMPARY: Cuda Multiple Precision Arithmetic Library and Applications. In *Mathematical Software – ICMS 2016, the 5th International Conference on Mathematical Software*, pages 232–240, Springer-Verlag, 2016.

convergence and performance

Two concerns when running Newton's method:

1. For which inputs does the method converge?
2. Which kernels occupy most of the running time?

Nearby singularities are problematic for convergence.

Theorem (the ratio theorem, Fabry 1896)

If for the series $x(t) = c_0 + c_1 t + c_2 t^2 + \cdots + c_n t^n + c_{n+1} t^{n+1} + \cdots$,
we have $\lim_{n \to \infty} c_n / c_{n+1} = z$, *then*

- z *is a singular point of the series, and*
- *it lies on the boundary of the circle of convergence of the series.*

Then the radius of this circle equals $|z|$.

The ratio c_n/c_{n+1} is the pole of Padé approximants of degrees $[n/1]$ (n is the degree of the numerator, with linear denominator).

Outline

1. Problem Statement
   - numerical analytic continuation
   - the need for multiple double precision
   - software packages QDlib, GQD, and CAMPARY
   - convergence and performance

2. Newton for Taylor Series
   - monomial and binomial homotopies
   - linearization of power series
   - staggered computations
   - computational results

monomial and binomial homotopies

Consider n variables $\mathbf{x}$, A is an n-by-n exponent matrix, and $\mathbf{b}(t)$ is a vector of n series of order $O(t^d)$:

$$\mathbf{x}^A = \mathbf{b}(t) \quad \text{is a } \textit{monomial homotopy}.$$

For example, $n = 3$, $\mathbf{x} = [x_1, x_2, x_3]$:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{cases} x_1 = b_1(t) & x_1(t) = \exp(\alpha_1 t) + O(t^d) \\ x_1 x_2 = b_2(t) & x_2(t) = \exp(\alpha_2 t) + O(t^d) \\ x_1 x_2 x_3 = b_3(t) & x_3(t) = \exp(\alpha_3 t) + O(t^d) \end{cases}$$

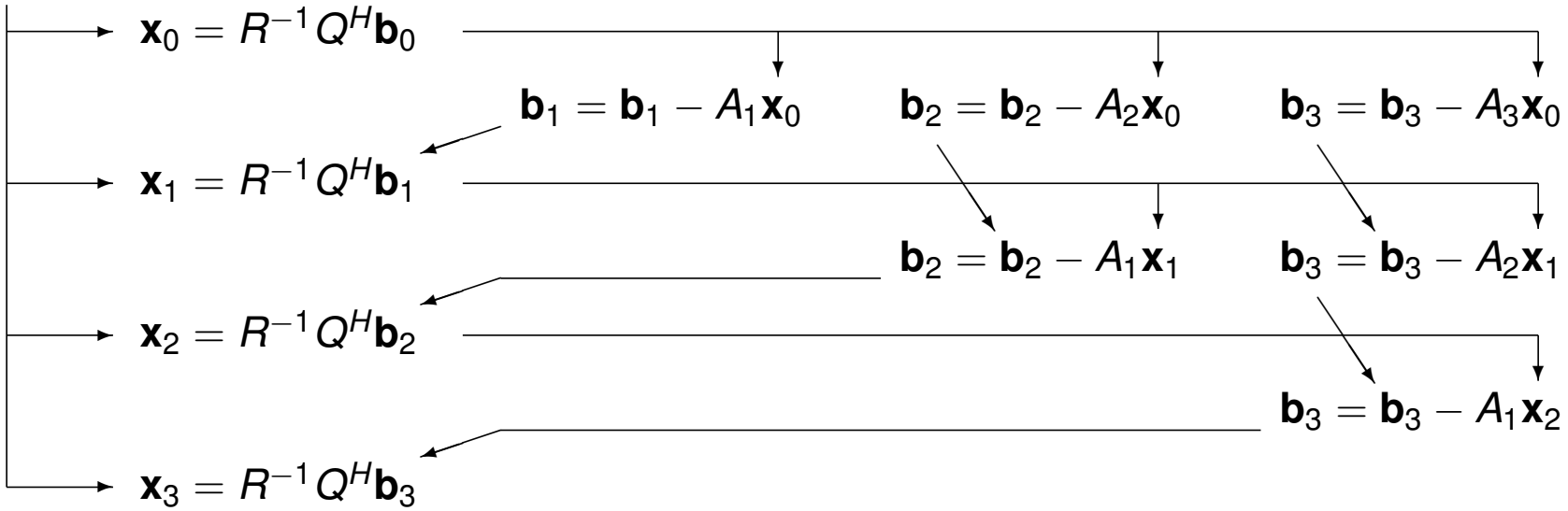
where $\exp(\alpha t) + O(t^4) = 1 + \alpha t + \frac{\alpha^2}{2!} t^2 + \frac{\alpha^3}{3!} t^3 + O(t^4)$,

with $\alpha \in [-1, -1+\delta] \cup [1-\delta, 1]$, $\delta > 0$, or $|\alpha| = 1$ for random $\alpha \in \mathbb{C}$.

$$\mathbf{c}_A \mathbf{x}^A + \mathbf{c}_{A^T} \mathbf{x}^{A^T} = \mathbf{b}(t) \quad \text{is a } \textit{binomial homotopy}.$$

a task graph for a triangular block Toeplitz system

$Q, R = \mathrm{qr}(A_0)$



$$\begin{bmatrix} A_0 & & & \\ A_1 & A_0 & & \\ A_2 & A_1 & A_0 & \\ A_3 & A_2 & A_1 & A_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

linearizes $A(t)\mathbf{x}(t) = \mathbf{b}(t)$, with

$$\begin{aligned} A(t) &= A_0 + A_1 t + A_2 t^2 + A_3 t^3, \\ \mathbf{x}(t) &= \mathbf{x}_0 + \mathbf{x}_1 t + \mathbf{x}_2 t^2 + \mathbf{x}_3 t^3, \quad \mathbf{b}(t) = \mathbf{b}_0 + \mathbf{b}_1 t + \mathbf{b}_2 t^2 + \mathbf{b}_3 t^3. \end{aligned}$$

different types of accelerated computations

1. convolutions for evaluation and differentiation
2. Householder QR
3. $Q^H\mathbf{b}$ computations
4. back substitutions to solve $R\mathbf{x} = Q^H\mathbf{b}$
5. updates $\mathbf{b} = \mathbf{b} - A\mathbf{x}$
6. residual computations $\|\mathbf{b} - A\mathbf{x}\|_1$

Which of the six types occupies most time?

staggered computations

Computing $\mathbf{x}(t) = \mathbf{x}_0 + \mathbf{x}_1 t + \mathbf{x}_2 t^2 + \cdots + \mathbf{x}_{d-1} t^{d-1}$, observe:

1. Start $\mathbf{x}_0$ with half its precision correct,
   otherwise Newton's method may not converge.

2. Increase d in the order $O(t^d)$ gradually,
   e.g.: the new d is $d + 1 + d/2$,
   hoping (at best) for quadratic convergence.

3. Once $\mathbf{x}_k$ is correct, the corresponding $\mathbf{b}_k = 0$,
   as $\mathbf{b}_k$ is obtained by evaluation, and then the update $\Delta\mathbf{x}_k$ should no longer be computed because

$$QR\Delta\mathbf{x}_k = \mathbf{b}_k = \mathbf{0} \quad \Rightarrow \quad \Delta\mathbf{x}_k = \mathbf{0}.$$

   This gives a criterion to stop the iterations.

graphics processing units

The code was developed for the "Volta" V100 NVIDIA GPU, and tested on the "Pascal" P100 and RTX 2080, 4080 NVIDIA GPUs.

NVIDIA GPU	CUDA	#MP	#cores/MP	#cores	GHz
Pascal P100	6.0	56	64	3584	1.33
Volta V100	7.0	80	64	5120	1.91
GeForce RTX 2080	7.5	46	64	2944	1.10
GeForce RTX 4080	8.9	58	128	7424	2.10

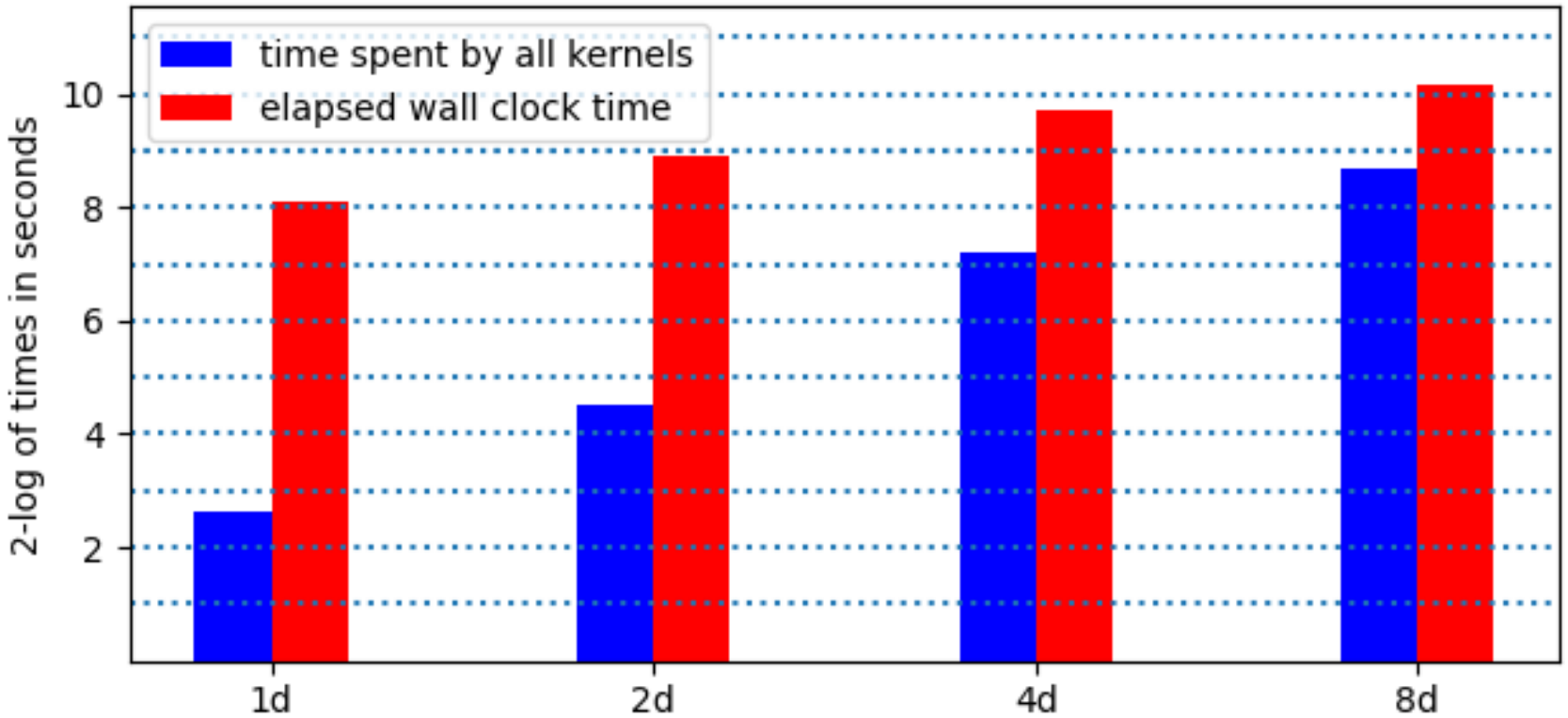
The double precision peak performance of the P100 is 4.7 TFLOPS. At 7.8 TFLOPS, the V100 is 1.66 times faster than the P100.

To evaluate the algorithms, compare the ratios of the wall clock times on the P100 over V100 with the factor 1.66.

For every kernel, the number of arithmetical operations is accumulated. The total number of double precision operations is computed using the cost overhead multipliers.

doubling precisions, wall clock and kernel times

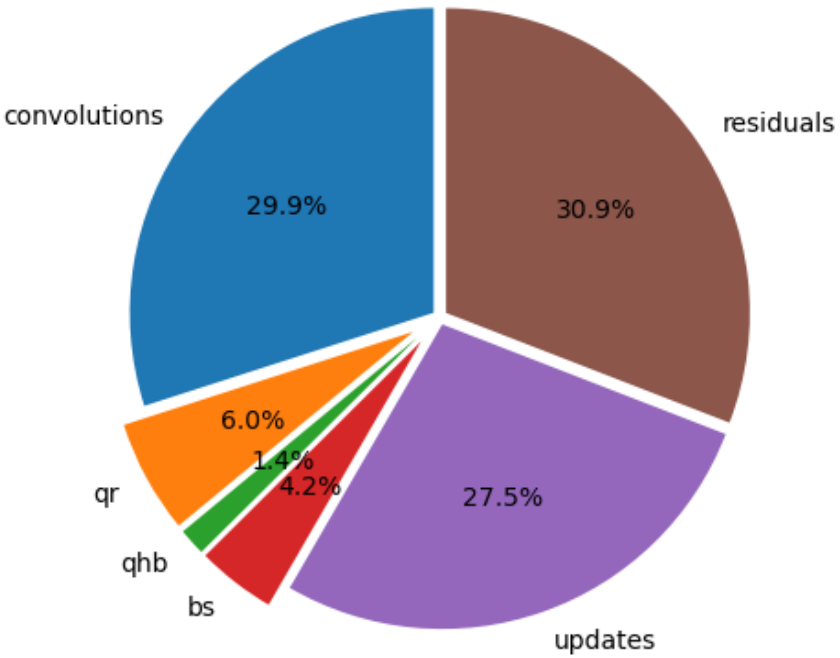
On one column of monomials, triangular exponent matrix of ones, $n = 1024$, $d = 64$, 24 steps, in 4 different precisions, on the V100:

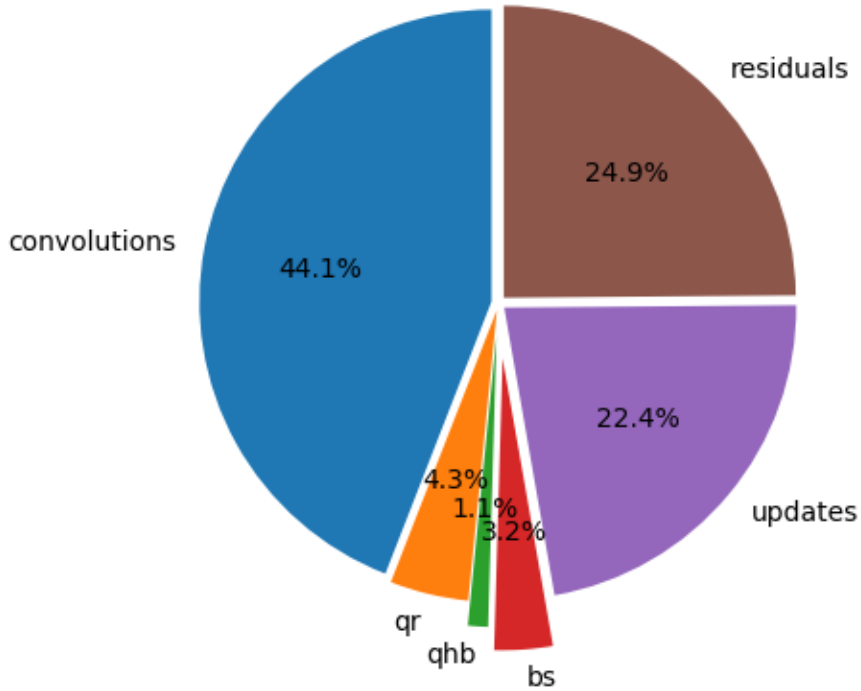


Doubling the precision less than doubles the wall clock time and increases the time spent by all kernels.

$n = 1024$, $d = 64$, 24 steps in octo double precision

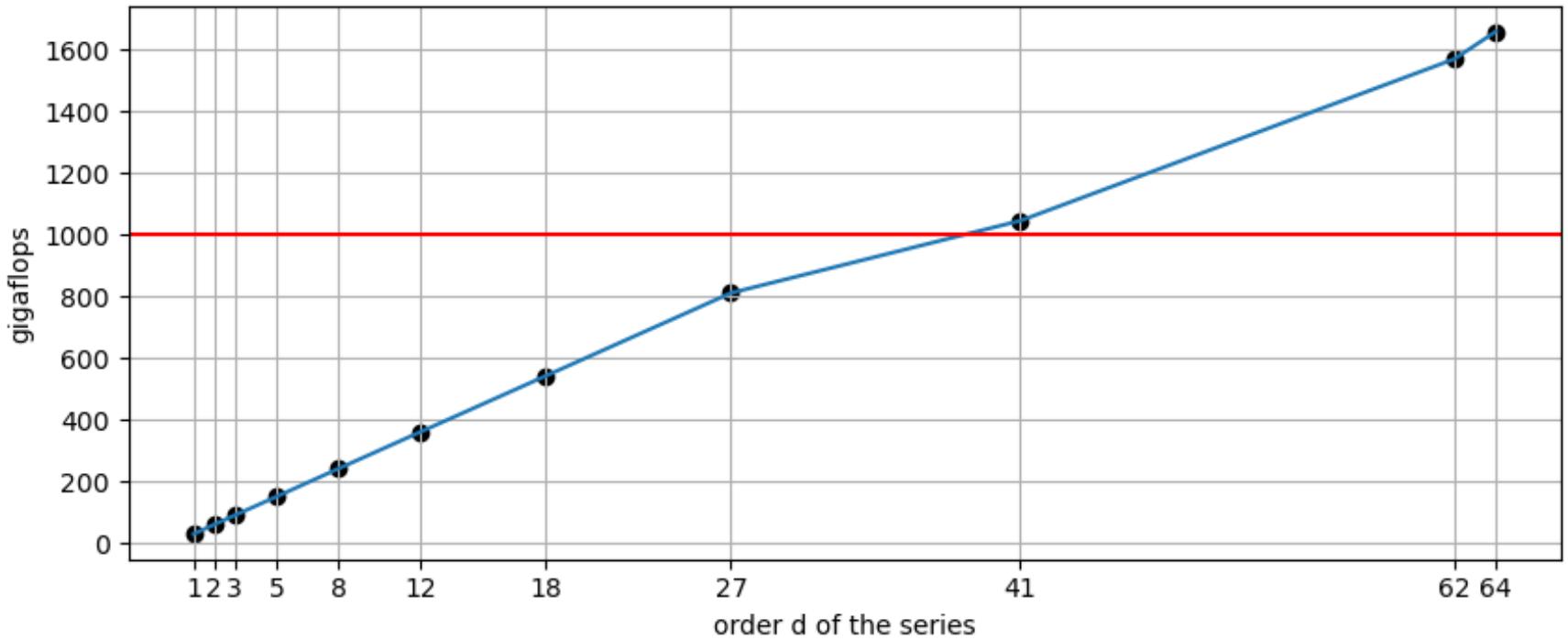
Six different types of accelerated computations, on the V100,
on one column of monomials and two columns of monomials:





teraflop performance of convolutions

On one column of monomials, triangular exponent matrix of ones,
$n = 1024$, performance of the evaluation and differentiation,
in octo double precision, for increasing orders of the series:



After degree 40, teraflop performance is observed on the V100.

two publications and a preprint

- **Accelerated polynomial evaluation and differentiation at power series in multiple double precision.**
  In the *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 740–749. IEEE, 2021.
  `arXiv:2101.10881`

- **Least squares on GPUs in multiple double precision.** In the *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 828–837. IEEE, 2022.
  `arXiv:2110.08375`

- **GPU Accelerated Newton for Taylor Series Solutions of Polynomial Homotopies in Multiple Double Precision.**
  `arXiv:2301.12659`

GPL-3.0 License, code at `github.com/janverschelde/PHCpack`